Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström

Göteborg 25 May. 2018

Examination in

ALGORITHMS TIN093 / DIT602

for D/IT/GU and others

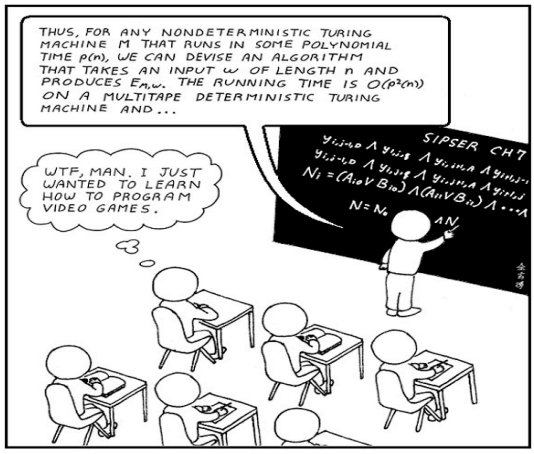
DAY: Wednesday DATE: 2018-05-30 TIME: 8.30-12.30 ROOM: M

Responsible teacher:	Erland Holmström tel. 1007, 0708-710600
Results:	Are sent by mail from Ladok.
Solutions:	Are eventually posted on homepage.
Inspection of grading:	The exam can be found in our study expedition after posting of results. Time for exam review are announced on homepage after the results are published. Or send me a mail so we can agree on a time.
Grade limits:	CTH: 3=28p, 4=38p, 5= 48p, GU: G=28p VG=48p, Phd: G=38p
Aids on the exam:	Handwritten A4 paper, both sides, dictionary. (Note that aids vary so check this before every exam.)

Observe:

- Write legible! Draw figures. Solutions that are difficult to read are not evaluated!
- Answer concisely and to the point. Write in English.
- All answers must be motivated.
- The advice and directions given during course must be followed.
- Start every new problem on a new sheet of paper.
- Write your exam-number on every paper.
- Start by reading thought all questions so you can ask questions when I come. I usually will come after ca 2 ours.
- Use pseudocode, indent properly, use comments and so on.

Good Luck!



How to describe algorithms

1) **Explain how it works**

Use words, high level pseudocode and illustrations.

The important thing is to reveal the underlying idea and as a help to understand the pseudocode in the next step. A "dry swim" of some steps is very good here.

2) **Give an abstract algorithm in pseudocode**

This is a mixture of programming language, mathematics, english, swedish and other suitable notation. You can use abstract data types (for instance graphs, sets, lists...) and sentences like "for every node w on EL(v) loop"

3) **Prove that it works**

Common techniques for this is induction proof, proof by contradiction, greed stays ahead and greedy exchange.

4) **Analyze its complexity**

It's not enough to say what it is, you must *motivate* the complexity from your algorithm.

It's not enough to analyze the problem, for instance to say that "since we are traversing a tree it must be O(the number of nodes in the tree)". This only gives a lower bound. You must analyze your algorithm.

5) **Describe the different parts** of your algorithm if that is needed for step 4 or 3

(To show that your algorithm can be implemented and to be able to analyze it in step 4.)

Ex: How do you pick the smallest of …, how is the graph implemented, how do you implement the sorting step, how to implement a subroutine you are using and so on.

6) **Implement more details** (only in lab assignment)

During this course you are expected to do step 1+2+3+4(+5) where (+5) is needed only if needed in step 4 or 3.

(So it doesn't matter if the question is "describe an algorithm for..., solve the problem..., find an...., or something else it is always 1+2+3+4(+5).)

Exceptions to this are always explicitly stated like "you only need to do step xx" or "only prove that …" or "analyze the complexity of the given algorithm" or clearly obvious like in "How is the class of NP-problems defined?"

Problem 1. T*est: warm up* (Motivations needed as usual.)

a) Tell me two things that influence the optimality of Greedy algorithms?

b) In order to use Dynamic programming, four requirements needs to be fulfilled. Tell me the two most important?

c) Describe the 4 problem classes P, NP, NPC and intractable (how are they *defined*).

How do you find out if a problem A belongs to the NPC class?

d) Is it the case that Interval Scheduling $\leq_P$ Vertex Cover?

e) Is it the case that Independent Set $\leq_P$ Interval Scheduling?

(10p)

Problem 2. *Tests: Induction proofs*

Give a quick sketch of quicksorts main part (assuming that partition and pivot works) and then use induction to prove that quicksort works.

(8p)

Remember: There are many problems but not that many solutions.....

Problem 3. *Test: dynamic programming*.

Let us imagine four economic regions I, II, III, and IV, in which it has been decided to undertake a sales promotion. A certain sum A is available for allocation among the four regions. It is postulated that the profits which can be realized in each region are known as a function of the investments, `p(region, inv)`.

If we assume that this sum A is equal to 100 millions counting in tens of millions then `p` could look like the table to the right.

Thus a policy would consist of allocating, for example, 30 in I, 10 in II, 50 in III, and 10 in IV. From the table, the profit would be 65+25+62+20= 172 millions. Notice that this is more than if we had invested 100 millions in region I.

Invest-ments	Reg. I	Reg. II	Reg. III	Reg. IV
10	28	25	15	20
20	45	41	25	33
30	65	55	40	42
40	78	65	50	48
50	90	75	62	53
60	102	80	73	56
70	113	85	82	58
80	123	88	90	60
90	132	90	96	60
100	138	90	100	60

If we generalize: Assume that you have an amount of capital, x dollars, that you want to invest in n regions. The smallest amount that can be invested in a region is 100 dollars. You may assume the existence of the function `p(region, inv)` that given a region and an investment returns the profit that will be achieved for that region/investment.

a) Describe an efficient recursive function, `invest(i, x)`, that computes the maximum return when x dollars are invested among the regions $1..i$. You don't need to prove that it works but motivate the recursive formula.

b) How do you find out how much to invest in each company?

c) Is your algorithm a polynomial algorithm? If not, what is it? Explain your answer.

(14p)

Problem 4. *Tests: backtracking, complexity*

Checking passwords of zeroes and ones: The computer specialist has left your company and he left without giving you his password to the computer system. Since he was the only one with root privileges this is a problem.

It was well known that he only used 0 and 1 to construct his password and that he changed it once a week so it seem to be a simple problem to solve. After trying for a while you realize that it could be harder than you think, after all you don't even know how many zeroes and ones he used, so you decide to write a program that does the job.

a) Write a program, `void perms(boolean[] x, int k)`, that generates all possible passwords of zeroes and ones with length *n = length of x* and test if they are the password. You don't need to prove that it works but motivate the recursive formula. (This program will have to be called with increasing lengths of x to find the password but you don't need to write that part)

   You may assume the existence of a method `boolean tryPassw(boolean[] x)` that given an array x with the zeroes and ones represented with false and true returns true if x contains the correct password, false otherwise. Complexity O(n).

   You may also assume the existence of a method `print(boolean[] x)` with the obvious functionality. Complexity O(n).

b) How many passwords are there as a function of n = the length of x (i.e. how many solutions will a call to your method `perms` above test in the worst case).

c) Since you are interested in knowing how long the program will take, you analyse its complexity. Do that, i.e set up the recursion equation, T(n)= …, and solve it.

(13p)

Problem 5. *Test: algorithm design, greed, proofs.*

chapters	pages	Imp.
1	120	5
2	150	5
3	200	4
4	150	8
5	140	3

You and your team has written a draft report that has 5 chapters. You have a lot to say but the number of pages in the report is maximized to 600 pages and you have written more than that. The figure shows the lengths of the chapters and their importance, where the scale is 1 (low) to 10 (high). The problem is to edit the report by deleting pages or rewriting so that the overall importance is maximized within these 600 pages.

For example, we could delete chapter 5 and 20 pages of chapter 3 to obtain a 600-page report (120+150+180+150+0= 600). The overall importance of this draft is 5+5+(180/200)* 4+8 = 21.6. [Chapter 3's importance is 4, but since we deleted 20 of its pages, it contributes only (180/200)*4 to the sum. Since we deleted chapter 5, it contributes zero to the sum.]

In general, if x_i is the amount of chapter i used (e.g., x3 180/200) and p_i is the importance of chapter i (e.g., p3=4), we define the importance of the resulting draft to be $\sum_{i=1}^{n} x_i p_i$

The problem, then, is to maximize the preceding sum subject to the condition that the report must consist of at most 600 pages. Write the program.

What is maximized for the problem if p_i =1 for all i?

15p)